# HÀM

## HÀM

### Khái niệm

Hàm trong C là một tập hợp các tác vụ xử lý tạo thành một khối nhằm thực hiện một chức năng cụ thể nào đó trong chương trình. Sau khi tạo và đặt tên cho một hàm, chúng ta chỉ cần gọi tên hàm ra mỗi khi cần sử dụng đến nó trong chương trình.

**Ưu điểm**: Bằng cách kết hợp các xử lý lại thành một hàm, chúng ta không cần viết lại các xử lý nhiều lần, qua đó có thể giảm sai sót khi viết code, cũng như có thể tái sử dụng hàm cho một chương trình khác.

### Cấu trúc hàm trong C

Khi học về hàm trong C chúng ta cần nắm rõ cấu trúc hàm gồm 3 thành phần chính, đó là **tham số**, **đối số** và **giá trị trả về**.

#### Cú pháp khai báo hàm trong C

Khi muốn sử dụng một hàm C do người dùng định nghĩa, trước hết chúng ta cần khai báo hàm trong C.

Chúng ta viết **kiểu dữ liệu trả về** từ hàm trước, sau đó đến **tên hàm**, rồi kiểu và tên của các **tham số truyền vào** sử dụng trong hàm, và cuối cùng là các **xử lý trong hàm** nằm giữa cặp ngoặc nhọn {} cùng với**giá trị trả về**, với cú pháp hàm như sau:

|  |
| --- |
| < Kiểu dữ liệu trả về > < Tên Hàm > (Tham số truyền vào){     Câu lệnh 1 trong hàm;     Câu lệnh 2 trong hàm;     ...     return < Giá trị trả về >;  } |

Trong đó:

* **Kiểu dữ liệu trả về** là kiểu dữ liệu của giá trị trả về từ hàm, ví dụ như int, char. Nếu hàm không trả về giá trị thì chúng ta cần chỉ định kiểu **[void]** để thay thế.
* **Tên Hàm** là một chuỗi ký tự dùng để đặt tên đại diện cho hàm.
* **Tham số** là các biến sử dụng trong khai báo hàm con, cũng như để nhận đối số (các giá trị truyền vào) khi chúng ta gọi hàm. Tham số có thể có hoặc không
* **return** là từ khóa dùng để trả giá trị trả về từ hàm trong C
* **Giá trị trả về** là các đối tượng được trả về sau khi hàm được thực thi. Ví dụ như hàm có thể trả về một số, một chuỗi, hoặc một con trỏ v.v..

|  |
| --- |
| **VD**: Chương trình nhập a, b và tính tổng ( sử dụng hàm con) |
| #include <stdio.h>  int Tinh\_Sum(int a, int b);  // Hàm con  int Tinh\_Sum(int a, int b) {      int sum = a + b;      return sum;  }  int main() {      int a, b, result = 0;      printf("Nhap a: ");      scanf("%d", &a);      printf("Nhap b: ");      scanf("%d", &b);      result = Tinh\_Sum(a, b); // Tinh\_Sum(a, b) là lời gọi hàm. a, b là các đối số      printf("Tong: %d", result);  } |

* Xét hàm con Tinh\_Sum.

|  |
| --- |
| int Tinh\_Sum(int a, int b) {      int sum = a + b;      return sum;  } |

Trong đó:

* int là kiểu dữ liệu trả về, kiểu dữ liệu trả về này phải cùng loại với biến cần trả về.
* Tinh\_Sum là tên hàm con.
* int a, int b là các Tham số. Các Tham số này phải cùng loại với Đối số
* Biến sum là biến trả về.
* Xét gọi hàm Tinh\_Sum.

|  |
| --- |
| result = Tinh\_Sum(a, b); |

Trong đó:

* Tinh\_Sum là tên hàm con cần gọi.
* a, b là các đối số.

#### Gọi hàm

Sau khi khai báo hàm trong C, chúng ta có thể sử dụng hàm đó nhiều lần trong chương trình bằng cách **gọi hàm trong C** với cú pháp sau đây:

|  |
| --- |
| Tên Hàm( đối số 1 , đối số 2 ,...); |

Chúng ta viết tên hàm, sau đó đặt các đối số (các giá trị truyền vào khi gọi hàm) ở giữa cặp dấu ngoặc đơn **()** và cách nhau bởi dấu phẩy **,** . Các giá trị của đối số sẽ được dùng để truyền tham số trong C.

Ví dụ, chúng ta khai báo và gọi hàm trong C như sau:

|  |
| --- |
| #include <stdio.h>  int find\_sum(int a,int b){     int sum = a + b;     return sum;  }  int main(){     find\_sum(1,2);     find\_sum(5,6);     return 0;  } |

**Lưu ý**: chúng ta gọi các biến dùng để khai báo hàm là **tham số**, còn các giá trị thực truyền vào khi gọi hàm là **đối số**. Giá trị của các đối số sẽ được các tham số tương ứng nhận và thực hiện các xử lý ở bên trong hàm.

Việc các **tham số bên trong hàm** nhận các **đối số từ bên ngoài hàm** được gọi là **truyền tham số trong C**. Chi tiết về tham số và đối số trong C sẽ được tìm hiểu tiếp tục sau đây

Sau khi gọi hàm trong C, một giá trị là kết quả của hàm sẽ được trả về. Giá trị trả về từ hàm có thể được gán vào biến để sử dụng như sau:

|  |
| --- |
| #include <stdio.h>  int find\_sum(int a,int b){     int sum = a + b;     return sum;  }  int main(void){     int ketqua = find\_sum(1,2);     printf("%d",ketqua);     return 0;  }  // 3 |

Trong trường hợp trong hàm được khai báo không chứa tham số, chúng ta cũng không thể truyền tham số vào hàm được. Khi đó chúng ta sẽ chỉ gọi hàm trong C bằng cách lược bỏ đối số đi như sau:

|  |
| --- |
| tên-hàm(); |

**VD:**

|  |
| --- |
| #include <stdio.h>  void say\_hi(){     printf("Hi!");  }  int main(void){     say\_hi();     return 0;  }  // Hi! |

#### Tham số và đối số trong C

##### Tham số là gì ?| Parameter

**Tham số (parameter)** là các biến có kiểu và tên được sử dụng để nhận các giá trị truyền vào (Đối số- argument) để xử lý trong hàm. Tham số được sử dụng khi chúng ta **khai báo hàm trong C**.

Các tham số sử dụng trong hàm đều cần chỉ định **kiểu và tên** của nó. Và **kiểu, tên** của tham số cần phải giống với **kiểu, tên** của loại đối tượng mà nó sẽ nhận khi đối tượng đó được **truyền vào hàm.**

Ví dụ, nếu bạn muốn truyền giá trị là một số thuộc **kiểu int**, thì bắt buộc là tham số khai báo trong hàm cũng phải thuộc **kiểu int.** Tương tự nếu đối tượng truyền vào là một **char, hay một chuỗi,** bạn cũng cần phải chỉ định kiểu **char hay char[]** tương ứng.

**VD:**

Các biến a,b sử dụng trong hàm được gọi là tham số, và chúng ta sử dụng **tham số** để khai báo hàm trong C

|  |
| --- |
| int find\_sum (int a, int b){     int sum = a + b;     return sum;  } |

##### Đối số là gì ? | Argument

**Đối số (argument)** là các giá trị thực được truyền vào khi **gọi hàm**. Đối số được truyền vào hàm qua sẽ được gán vào tham số, và được sử dụng khi chúng ta **gọi hàm trong C**.

Các giá trị thực truyền vào khi ta gọi hàm được gọi là **đối số**, và chúng ta dùng đối số để truyền tham số trong C

**VD:**

|  |
| --- |
| int find\_sum (int a, int b){     int sum = a + b;     return sum;  }  int main(){     int tong = find\_sum(1 , 2);  } |

Có nhiều loại giá trị có thể truyền vào hàm khi chúng ta gọi hàm, ví dụ như biến, con trỏ và cấu trúc chẳng hạn. Với mỗi loại đối tượng như vậy sẽ có kiểu dữ liệu khác nhau, và chúng ta cần cần chú ý phải truyền đối tượng vào hàm có kiểu dữ liệu giống với kiểu của tham số tương ứng trong hàm dùng để nhận nó.

Ví dụ nếu khi khai báo hàm, chúng ta sử dụng một tham số có kiểu int, thì chúng ta cũng chỉ có thể truyền đối số có cùng kiểu int vào hàm mà thôi. Nếu không thì hàm không chạy được và lỗi sẽ bị xảy ra.

Trong khi gọi một hàm, có hai cách trong đó các đối số có thể được chuyển đến một hàm:

|  |  |
| --- | --- |
| **Cách** | **Kiểu gọi & mô tả** |
| 1 | [**Call by value**](https://viettuts.vn/lap-trinh-c/call_by_value)  Phương thức này sao chép giá trị thực của một đối số vào tham số chính thức của hàm. Trong trường hợp này, các thay đổi được thực hiện cho tham số bên trong hàm không ảnh hưởng đến đối số. |
| 2 | [**Call by reference**](https://viettuts.vn/lap-trinh-c/call_by_reference)  Phương thức này sao chép địa chỉ của một đối số vào tham số chính thức. Bên trong hàm, địa chỉ được sử dụng để truy cập đối số thực tế được sử dụng trong cuộc gọi. Điều này có nghĩa là các thay đổi được thực hiện cho tham số ảnh hưởng đến đối số. |

##### Sự khác biệt giữa một đối số và một tham số

Tổng kết sự khác nhau giữa **đối số** và **tham số**.

* Tham số là các biến có tên được sử dụng trong khai báo hàm.
* Đối số của hàm là các giá trị thực được truyền vào khi gọi hàm.
* Tham số nhận giá trị của các đối số được truyền vào từ ngoài hàm và thực hiện tính toán bên trong hàm.

#### Giá trị trả về trong C

##### Return trong C là gì

**Return trong C** là một từ khóa (keyword) dùng để trả về giá trị từ hàm. Return có tác dụng kết thúc hàm và trả lại điều khiển cũng như kết quả xử lý hàm cho người gọi. Chúng ta có thể sử dụng hoặc lược bỏ return khi khai báo hàm trong C, và một hàm có chứa return trong C được gọi là hàm trả về giá trị trong C.

|  |
| --- |
| int find\_sum (int a, int b){     int sum = a + b;     return sum;  }  **Giá trị trả về** là một từ khóa để trả về giá trị từ hàm. Chúng ta sử dụng return khi khai báo hàm trong C |

##### Cách dùng return trong C

Tùy thuộc vào việc hàm có trả về giá trị hay không mà chúng ta có những cách sử dụng return trong C khác nhau.

##### return trong hàm trả về giá trị

Khi sử dụng return trong hàm trả về giá trị trong C, chúng ta viết giá trị trả về của hàm đằng sau lệnh return như sau:

|  |
| --- |
| <Kiểu dữ liệu trả về> <Tên Hàm> ( kiểu 1 tham số 1, kiểu 2 tham số 2, ...){     Câu lệnh;     Câu lệnh;     return giá-trị-trả-về;  } |

**VD:**

|  |
| --- |
| int find\_sum(int a,int b){     int sum = a + b;     return sum;  }  int main(void){     int ketqua = find\_sum(1,2);     printf("%d",ketqua);     return 0;  } |

**Kết quả:**

|  |
| --- |
| **3** |

##### return trong hàm không trả về giá trị

Trong hàm không trả về giá trị trong C, chúng ta lược bỏ lệnh return, và chỉ định **kiểu dữ liệu trả về** thành kiểu **void** như sau:

|  |
| --- |
| void Tên Hàm( kiểu 1 tham số 1 , kiểu 2 tham số 2, ...){     Câu lệnh;     Câu lệnh;  } |

**VD:**

|  |
| --- |
| void find\_sum(int a,int b){     int sum = a + b;     printf("%d", sum);  }  int main(void){     find\_sum(1,2);     return 0;  } |

**Kết quả:**

|  |
| --- |
| **3** |

##### return trả về kết quả biểu thức

Ở phần trên chúng ta đã biết cách dùng return để trả về một giá trị cụ thể rồi. Tuy nhiên ngoài cách trả về một giá trị cụ thể như một số, một ký tự v.v.. như vậy thì chúng ta cũng có thể sử dụng return để trả về kết quả của một biểu thức tính toán.

Ví dụ, chúng ta có thể trả về kết quả phép cộng bằng cách viết trực tiếp biểu thức tính toán sau lệnh return như sau:

|  |
| --- |
| #include <stdio.h>  int find\_sum(int a,int b){     return a + b;  }  int main(void){     int ketqua = find\_sum(1,2);     printf("%d",ketqua);     return 0;  } |

**Kết quả:**

|  |
| --- |
| **3** |

##### Hàm trả về nhiều giá trị trong C

##### Hàm trả về nhiều giá trị trong C

Ngoài việc trả về giá trị, câu lệnh return cũng được sử dụng để **kết thúc quá trình xử lý của hàm**. Bằng cách sử dụng return, bạn có thể kết thúc một hàm tại một thời điểm nào đó khi đã thoả mãn một điều kiện ban đầu.

Ví dụ, chúng ta có thể kết thúc một hàm tuỳ thuộc vào giá trị nhập vào hàm đó như sau:

|  |
| --- |
| #include <stdio.h>    char \*checkNum(int num) {     if (num > 10) {        return "Gia tri qua lon";     }       if (num < 1) {        return "Gia tri qua nho";     }       return "Gia tri trong pham vi tu 1 den 10";  }    int main() {     int num;       num = 3;     printf("num = %d, %s\n", num, checkNum(num));       num = 30;     printf("num = %d, %s\n", num, checkNum(num));       num = -3;     printf("num = %d, %s\n", num, checkNum(num));       return 0;  } |

**Kết quả:**

|  |
| --- |
| num = 3, Gia tri trong pham vi tu 1 den 10  num = 30, Gia tri qua lon  num = -3, Gia tri qua nho |

Bạn có thể thấy tuỳ thuộc vào giá trị truyền hàm vào, mà hàm có thể được kết thúc tại theo các điều kiện khác nhau, bằng cách sử dụng câu lệnh return như trên.

##### return 0 và return 1 trong hàm main()

return 0 và return 1 là hai giá trị trả về duy nhất của hàm main() trong C.

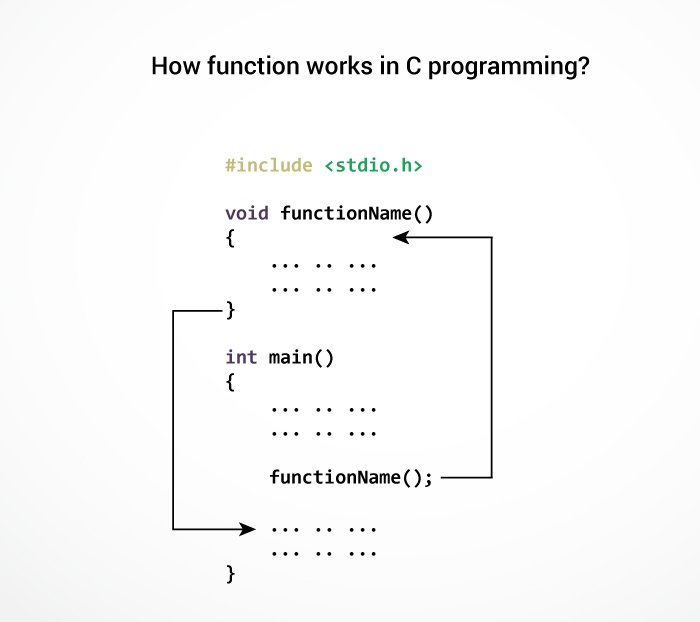
Hai giá trị trả về này của hàm main() có ý nghĩa như sau:

* Chúng ta chỉ đinh **return 0** để kết thúc chương trình theo cách bình thường (normal termination). Điều đó có nghĩa là kể cả chương trình có xảy ra lỗi hay không, thì C vẫn ngầm định là chương trình đã được kết thúc mà không có lỗi xảy ra.
* Chúng ta chỉ định **return 1** để kết thúc chương trình theo cách bất thường (abnormal termination). Điều đó có nghĩa là khi chương trình xảy ra lỗi, thì lỗi này sẽ được trả về khi kết thúc chương trình.

### Cách hoạt động của hàm trong C

Hình ảnh dưới đây cho bạn thấy cách hoạt động của hàm (chương trình con) ở trong ngôn ngữ C. Khi một lời gọi hàm được thực thi thì:

* Chương trình của bạn sẽ nhảy tới nơi định nghĩa hàm đó và thực thi các lệnh từ trên xuống dưới ở trong hàm đó.
* Khi hàm thực hiện xong, chương trình tiếp tục quay về thực hiện các lệnh phía sau lời gọi hàm.



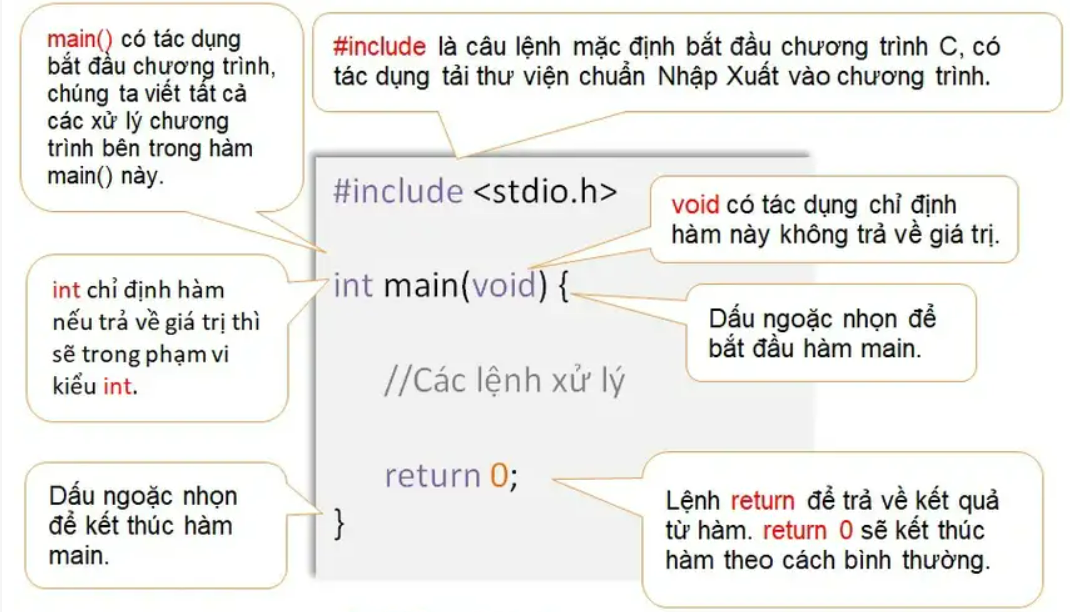
### Phân loại hàm trong C

Hàm được chia thành 3 loại:

* Hàm main
* Hàm đã được định nghĩa trong thư viện.
* Hàm do người dùng định nghĩa.

#### Hàm main trong C

Trong ngôn ngữ C, một chương trình là một tập hợp các hàm, với mỗi hàm trong chương trình là “tập hợp các quy trình” cần xử lý. Và trong các hàm đó thì **hàm main() trong C** là **hàm đầu tiên được thực thi** khi bắt đầu chạy một chương trình C.



#### Hàm được định nghĩa trong thư viện

Trong C cũng cấp sẵn cho người lập trình khá nhiều thư viện nhằm hỗ trợ thực hiện nhanh chóng một nhiệm vụ. Nếu như bạn đã từng học về ngôn ngữ C căn bản chắc hẳn đã quá quen thuộc với các thư viện: <stdio.h>, <stdlib.h>, <math.h>…ngoài ra, ngôn ngữ C còn cung cấp một số thư viện bổ xung ở bảng bên dưới:

|  |  |
| --- | --- |
| **Tên thư viện** | **Mô tả** |
| #include <assert.h> | Thư viện C cung cấp một macro gọi là assert có thể được sử dụng để kiểm tra một giả định được tạo bởi chương trình và in một thông báo chẩn đoán tìm lỗi nếu giả định này là false. |
| #include <ctype.h> | Thư viện xử lý và kiểm tra và ánh xạ các ký tự. |
| #include <errno.h> | Thư viện định nghĩa biến nguyên errno, được thiết lập bởi system call. |
| #include <float.h> | Thư viện cung cấp các hằng số liên quan đến giá trị số trong phạm vi số thực. |
| #include <limits.h> | Thư viện xác định các thuộc tính đa dạng của các kiểu biến khác nhau. |
| #include <locale.h> | Thư viện định nghĩa các thiết lập riêng về vị trí, chẳng hạn như định dạng date và các biểu tượng về currency |
| #include <math.h> | Thư viện cung cấp các hằng số và các hàm toán học. |
| #include <signal.h> | Thư viện định nghĩa một kiểu biến sig\_atomic\_t, hai lời gọi hàm, và một số macro để xử lý các tín hiệu khác nhau được báo cáo trong khi thực thi một chương trình. |
| #include <stdarg.h> | Thư viện định nghĩa một kiểu biến va\_list và 3 macro mà có thể được sử dụng để lấy các tham số trong một hàm khi số tham số là chưa được biết. |
| #include <stddef.h> | Thư viện chuẩn định nghĩa các kiểu biến và macro đa dạng. |
| #include <stdio.h> | Thư viện chuẩn định nghĩa 3 kiểu biến, một số macro và các hàm đa dạng để thực hiện input và output. |
| #include <stdlib.h> | Thư viện cung cấp các hàm được sử dụng để cấp phát bộ nhớ động. |
| #include <string.h> | Thư viện cung cấp các hàm xử lý chuỗi. |
| #include <time.h> | Thư viện cung cấp các hàm xử lý và thao tác với kiểu thời gian. |

Các hàm được tạo ra và thuộc trong một số thư viện chuẩn trong C nhằm mục đích tiện lợi hơn trong việc lập trình. Trong quá trình làm việc với C một cách căn bản, chúng ta đã từng gọi và sử dụng các hàm như **printf()** hay **scanf()** trong thư viện **<stdio.h>** để thực hiện các thao tác nhập xuất căn bản. Tuy nhiên, trong C còn cung cấp thêm nhiều các hàm nằm trong các thư viện chuẩn khác để sử dụng cho các mục đích khác nhau:

|  |  |
| --- | --- |
| **Tên hàm** | **Mô tả** |
| abort() | Hàm được sử dụng ngừng thực thi chương trình một cách bất thường. |
| abs() | Trả về trị tuyệt đối của một số nguyên đầu vào. |
| atexit() | Được sử dụng để gọi hàm func khi chương trình kết thúc |
| atof() | Chuyển đổi một chuỗi từ str sang số thực (double) |
| atoi() | Chuyển đổi một chuỗi từ str sang số nguyên (int) |
| atol() | Chuyển đổi một chuỗi từ str sang kiểu long int |
| calloc() | Cấp phát bộ nhớ được yêu cầu và trả về một con trỏ tới nó. |
| ldiv() | Chia tử số cho mẫu số |
| exit() | Kết thúc tiến trình đang gọi. |
| fclose() | Đóng file đã được mở. |
| ferror() | Kiểm tra lỗi khi mở file. |
| fflush() | Xóa bộ nhớ đệm đầu ra của một Stream. |
| fgetc() | Lấy ký tự kế tiếp (một unsigned char) từ Stream |
| fgetpos() | Lấy vị trí file hiện tại của Stream và ghi nó tới pos |
| fgets() | Đọc một dòng từ Stream đã cho và lưu trữ nó vào trong chuỗi được trỏ bởi str. |
| fopen() | Mở file theo chế độ định trước. |
| fprintf() | Gửi output đã được định dạng tới một Stream. |
| fputc() | Ghi một ký tự (một unsigned char) đã được xác định bởi tham số char tới Stream đã cho. |
| fputs() | Ghi một chuỗi tới Stream đã xác định. |
| fread() | Đọc dữ liệu từ Stream vào một mảng được trỏ tới. |
| free() | Giải phóng bộ nhớ đã được cấp phát trước đó bởi một lời gọi tới calloc, malloc, hoặc realloc. |
| fscanf() | Đọc input đã được định dạng từ một Stream. |
| fwrite() | Ghi dữ liệu từ mảng được trỏ bởi ptr tới Stream đã cho. |
| getc() | Lấy ký tự kế tiếp (một unsigned char) từ Stream đã cho |
| getchar() | Lấy một ký tự (một unsigned char) từ stdin |
| gets() | Đọc một dòng từ stdin và lưu trữ nó bên trong chuỗi được trỏ bởi str |
| malloc() | Cấp phát bộ nhớ được yêu cầu và trả về một con trỏ tới nó. |
| memcpy() | Sao chép n ký tự từ str2 tới str1. |
| memchr() | Tìm kiếm sự xuất hiện đầu tiên của ký tự c (một unsigned char) trong n byte đầu tiên của chuỗi được trỏ tới bởi tham số str. |
| memcmp() | So sánh n byte đầu của hai chuỗi str1 và str2. |
| memset() | Sao chép ký tự c (một unsigned char) tới n ký tự đầu tiên của chuỗi được trỏ tới bởi tham số str. |
| printf() | Gửi output đã được định dạng tới một stdout. |
| putc() | Ghi một ký tự (một unsigned char) được xác định bởi tham số char tới Stream đã cho. |
| putchar() | Ghi một ký tự (một unsigned char) đã được xác định bởi tham số char tới stdout. |
| puts() | Ghi một chuỗi str tới stdout (không ghi ký tự null). |
| remove() | Xóa filename đã cho để nó không thể truy cập được nữa. |
| rename() | Đổi tên file thành tên mới. |
| scanf() | Đọc input đã được định dạng từ stdin. |
| strchr() | Tìm kiếm sự xuất hiện đầu tiên của ký tự c (một unsigned char) trong chuỗi được trỏ tới bởi tham số str. |
| strcmp() | So sánh chuỗi được trỏ tới bởi str1 với chuỗi được trỏ tới bởi str2. |
| strcpy() | Sao chép chuỗi được trỏ tới bởi str2 tới str1. |
| strlen() | Độ dài của chuỗi str (không bao gồm ký tự null kết thúc). |
| strncat() | Nối chuỗi mới vào sau chuỗi ban đầu |

##### Các hàm có sẵn trong C

**Các hàm có sẵn trong C** là các hàm được chuẩn bị sẵn trong các thư viện chuẩn của C. Các hàm này được tích hợp bên trong các header file, ví dụ như hàm **strlen()** tích hợp trong header file **[string.h**] chẳng hạn.

Các hàm này mặc dù đã được chuẩn bị sẵn, nhưng chúng ta chỉ có thể thực thi chúng trong chương trình C, nếu chúng ta gọi chúng. Để sử dụng được các hàm có sẵn này, chúng ta cần phải thêm (include) các header file này vào đầu mỗi chương trình.

**VD:** Các hàm như hàm print(), hàm scanf() v.v.. đều là các hàm có sẵn trong tích hợp trong header file **[stdio.h],** hay hàm strlen() lại được tích hợp trong header file **[string.h]**, và chúng ta cần include các file này trong chương trình như sau:

|  |
| --- |
| #include <stdio.h>  #include <string.h>  int main(void){     printf("Nhap chuoi ky tu: ");       char str[100];     scanf("%s", str);       int len = strlen(str); //Hàm strlen dùng để tính độ dài chuỗi.     printf("Do dai chuoi: %d",len);     return 0;  } |

**Kết quả:**

|  |
| --- |
| Nhap chuoi ky tu: abc Do dai chuoi: 3 |

#### Hàm do người dùng định nghĩa

**Hàm C do người dùng định nghĩa** (user-defined functions)) là các hàm mà chúng ta tự mình tạo ra bằng cách khai báo chúng trong chương trình. Sau khi khai báo hàm, chúng ta có thể sử dụng chúng trực tiếp trong chương trình bằng cách gọi chúng tại bất kỳ thời điểm nào mà chúng ta muốn.

Hàm C do người dùng định nghĩa có ưu điểm rất lớn, đó là sự tự do điều chỉnh hàm theo ý mà bạn muốn, bởi vì hàm là do bạn tự tạo ra, tự bạn thiết kế và làm chủ nó.

##### Hàm không có tham số truyền vào, không có kiểu dữ liệu trả về

Hàm không có tham số truyền vào và không có kiểu dữ liệu trả về (void) thường được sử dụng để thực hiện các tác vụ không trả về kết quả, như xuất thông tin ra màn hình, đọc dữ liệu từ bàn phím, hoặc thực hiện một tác vụ nào đó nhưng không cần trả về bất kỳ giá trị nào.

Cấu trúc của hàm không có tham số truyền vào và không có kiểu dữ liệu trả về như sau:

|  |
| --- |
| void <Tên Hàm>() {      // khối lệnh của hàm  } |

Trong đó:

**void**: là kiểu dữ liệu của hàm không trả về giá trị.

**<Tên Hàm>:** là tên của hàm.

**khối lệnh của hàm:** là tập hợp các lệnh và biểu thức để thực hiện các tác vụ cần thiết.

Ví dụ về hàm không có tham số truyền vào và không có kiểu dữ liệu trả về để tính tổng hai số a và b:

|  |
| --- |
| #include <stdio.h>  void tong() {     int a = 5, b = 7;     int c = a + b;     printf("Tong hai so %d va %d la %d", a, b, c);  }  int main() {     tong(); // Gọi hàm sum để tính tổng hai số a và b     return 0;  } |

**Kết quả:**

|  |
| --- |
| **Tong hai so 5 va 7 la 12** |

Các bước thực hiện của ví dụ tính tổng hai số a và b bằng hàm không có tham số truyền vào và không có kiểu dữ liệu trả về như sau:

1. Khai báo hàm sum không có tham số truyền vào và không có kiểu dữ liệu trả về.
2. Trong hàm sum, khai báo hai biến nguyên a và b, và gán giá trị cho chúng là 5 và 7.
3. Tính tổng hai số a và b, lưu vào biến c.
4. Sử dụng hàm printf để in kết quả tổng hai số a và b ra màn hình.

5. Trong hàm main, gọi hàm sum để tính tổng hai số a và b.

##### Hàm không có tham số truyền vào, có kiểu dữ liệu trả về

Hàm không có tham số truyền vào, có kiểu dữ liệu trả về là một loại hàm trong ngôn ngữ lập trình C. Hàm này không yêu cầu bất kỳ tham số đầu vào nào, nhưng sẽ trả về một giá trị theo kiểu dữ liệu được xác định trước.

Cấu trúc của hàm không có tham số truyền vào, có kiểu dữ liệu trả về này như sau:

|  |
| --- |
| < Kiểu dữ liệu trả về > < Tên Hàm >() {      // Khối lệnh của hàm     return < giá trị trả về >;  } |

Trong đó:

**< Kiểu dữ liệu trả về >:** là kiểu dữ liệu của giá trị trả về của hàm.

**< Tên Hàm >:**  là tên của hàm.

**Khối lệnh của hàm** là tập hợp các lệnh thực hiện các công việc cần thiết.

**return** được sử dụng để trả về một giá trị có kiểu dữ liệu được xác định trước từ hàm. Giá trị này sẽ được truyền cho lời gọi hàm.

Ví dụ về hàm không có tham số truyền vào và có kiểu dữ liệu trả về để tính tổng hai số a và b:

|  |
| --- |
| #include <stdio.h>  int sum() {     int a = 5, b = 7;     int c = a + b;     return c;  }  int main() {     int tong = sum(); // Gọi hàm sum và lưu kết quả vào biến tong     printf("Tong hai so 5 va 7 la %d", tong);     return 0;  } |

**Kết quả:**

|  |
| --- |
| **Tong hai so 5 va 7 la 12** |

##### Hàm có tham số truyền vào, không có kiểu dữ liệu trả về

Hàm có tham số truyền vào, không có kiểu dữ liệu trả về là loại hàm trong ngôn ngữ lập trình C mà nhận vào một hoặc nhiều tham số, thực hiện một số tác vụ và không trả về bất kỳ giá trị nào.

Cấu trúc của một hàm có tham số truyền vào, không có kiểu dữ liệu trả về như sau:

|  |
| --- |
| void <Tên Hàm> ( kiểu 1 tham số 1 , kiểu 2 tham số 2, ...){     // khối lệnh của hàm  } |

Trong đó

**void**: là kiểu dữ liệu của hàm không trả về giá trị.

**<Tên Hàm>:** là tên của hàm.

**kiểu:** là kiểu dữ liệu của tham số truyền vào. Bạn có thể có nhiều tham số hơn nếu cần thiết, mỗi tham số phân cách nhau bằng dấu phẩy.

**tham số:** là một biến được định nghĩa trong phần khai báo của hàm, nhằm nhận giá trị truyền vào cho hàm khi được gọi

**khối lệnh của hàm:** là tập hợp các lệnh và biểu thức để thực hiện các tác vụ cần thiết.

Ví dụ về hàm có tham số truyền vào không có dữ liệu trả về để tính tổng 2 số a và b

|  |
| --- |
| #include <stdio.h>  void tong(int a, int b) {     int tong = a + b;     printf("Tong hai so %d va %d la %d.\n", a, b, tong);  }  int main() {     int a = 5;     int b = 7;     tong(a, b);     return 0;  } |

**Kết quả:**

|  |
| --- |
| **Tong hai so 5 va 7 la 12.** |

**Các bước thực hiện**

1. Khai báo hàm **tong** với hai tham số kiểu **int** tên là **a** và **b**, và không có giá trị trả về **(void)**.
2. Trong thân hàm **tong**, tính tổng của **a** và **b** bằng cách tạo một biến tong kiểu **int** và gán giá trị bằng **a + b**.
3. Sử dụng hàm **printf** để in ra màn hình thông tin về tổng hai số **a** và **b** vừa tính được.
4. Trong hàm main, khai báo và gán giá trị cho hai biến **a** và **b**.
5. Gọi hàm **tong** với tham số là **a** và **b.**

##### Hàm có tham số truyền vào, có kiểu dữ liệu trả về

Hàm có tham số truyền vào và có kiểu dữ liệu trả về là loại hàm cho phép chúng ta tính toán hoặc xử lý dữ liệu dựa trên giá trị được truyền vào qua các tham số và trả về kết quả sau khi hoàn thành xử lý.

Cấu trúc của hàm có tham số truyền vào có kiểu dữ liệu trả về bao gồm:

|  |
| --- |
| < Kiểu dữ liệu trả về > < Tên Hàm >(kiểu 1 tham số 1 , kiểu 2 tham số 2, ...) {      // Khối lệnh của hàm     return <giá trị trả về>;  } |

Trong đó

**< Kiểu dữ liệu trả về >:** là kiểu dữ liệu của giá trị trả về của hàm.

**<Tên Hàm>:** là tên của hàm.

**kiểu:** là kiểu dữ liệu của tham số truyền vào. Bạn có thể có nhiều tham số hơn nếu cần thiết, mỗi tham số phân cách nhau bằng dấu phẩy.

**tham số:** là một biến được định nghĩa trong phần khai báo của hàm, nhằm nhận giá trị truyền vào cho hàm khi được gọi

**khối lệnh của hàm:** là tập hợp các lệnh và biểu thức để thực hiện các tác vụ cần thiết.

Ví dụ về hàm có tham số truyền vào không có dữ liệu trả về để tính tổng 2 số a và b

**return** được sử dụng để trả về một giá trị có kiểu dữ liệu được xác định trước từ hàm. Giá trị này sẽ được truyền cho lời gọi hàm.

Ví dụ, chúng ta có thể viết một hàm tính tổng của hai số và trả về kết quả là giá trị tổng đó.

|  |
| --- |
| #include <stdio.h>  int tong(int a, int b) {     int tong = a + b;     return tong;  }  int main() {     int a = 5;     int b = 7;     int ket\_qua = tong(a, b);     printf("Tong hai so %d va %d la %d.\n", a, b, ket\_qua);     return 0;  } |

**Kết quả:**

|  |
| --- |
| **Tong hai so 5 va 7 la 12.** |

**Các bước thực hiện**

1. Khai báo hàm "tong" có hai tham số truyền vào là a và b, kiểu dữ liệu trả về là một số nguyên.
2. Trong hàm "tong", khai báo một biến tong và gán giá trị tổng của hai số a và b vào biến này.
3. Trong hàm "tong", sử dụng lệnh return để trả về giá trị của biến tong.
4. Trong hàm "main", khai báo hai biến nguyên a và b và gán giá trị cho chúng.
5. Trong hàm "main", gọi hàm "tong" với tham số a và b, lưu giá trị trả về vào biến ket\_qua.
6. In ra màn hình giá trị của a, b và ket\_qua bằng cách sử dụng lệnh printf.

## TRUYỀN THAM CHIỀU VÀ TRUYỀN THAM TRỊ

Một điều khá quan trọng trong C đó là **truyền tham chiếu và tham trị vào một phương thức (Function)**.

### Truyền tham chiếu

Truyền tham chiếu chính là cách chúng ta truyền cho nó một bản gốc thông qua địa chỉ ‘**&**‘. Nghĩa là giá trị của biến tham số truyền vào hàm, sẽ là giá trị của biến truyền vao [**hàm**](https://khuenguyencreator.com/ham-la-gi-cach-su-dung-ham-trong-lap-trinh-c/) đó.

**Nói đơn giản truyền tham chiếu là truyền địa chỉ.**

**Xét ví dụ:**

|  |
| --- |
| #include <stdio.h>  void incValue(int &x)  {      x++;      printf("Bien trong ham = %d \n", x);  }  int main ()  {      int a = 10;      incValue(a);      printf("Bien ngoai ham ham = %d \n", a);      return 0;  } |

**Kết quả:**

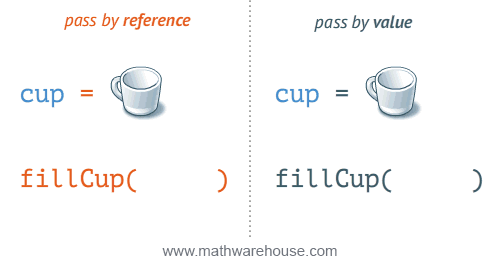
|  |
| --- |
| Bien trong ham = 11  Bien ngoai ham ham = 11 |

**Giải thích:**

Khi truyền biến a vào &x tức là a và x sẽ dùng chung địa chỉ. Mỗi khi thay đổi giá trị trên a hoặc x, thì giá trị của biến kia cũng thay đổi theo. Điều này giống việc một địa chỉ có 2 người thuê nhà. Bạn có thể truy vấn bằng tên người này hoặc người kia đều được.

Hàm scanf chính là một hàm tham chiếu chúng ta hay sử dụng, thực chất nó sẽ copy giá trị từ bàn phím vào địa chỉ của biến truyền vào mà thôi.

Khi sử dụng đối số tham chiếu xảy ra rủi ro về dữ liệu bị thay đổi không như ý muốn là rất cao.



*Ví dụ về truyền tham chiếu (pass by reference) và truyền tham trị (pass by value)*

### Truyền tham trị

Truyền tham trị là truyền cho đối số một bản sao, nghĩa là giá trị của biến đó sẽ được sao chép sang biến tham số truyền vào hàm. Đây cũng là cách chúng ta sử dụng hàm bình thường.

**Nói đơn giản truyền tham trị là truyền giá trị.**

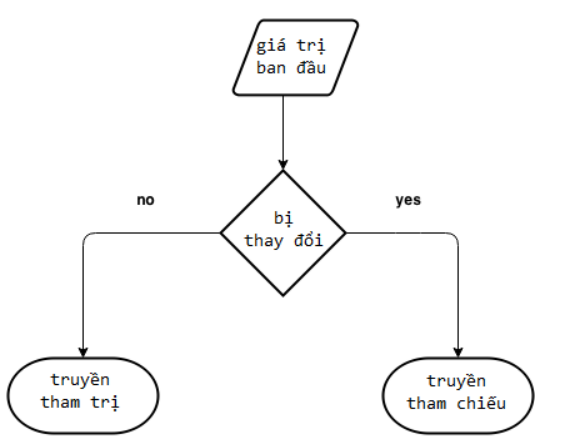
**Xét ví dụ:**

|  |
| --- |
| #include <stdio.h>  void incValue(int x)  {      x++;      printf("Bien trong ham = %d \n", x);  }  int main ()  {      int a = 10;      incValue(a);      printf("Bien ngoai ham ham = %d \n", a);      return 0;  } |

**Kết quả:**

|  |
| --- |
| **Bien trong ham = 11**  **Bien ngoai ham ham = 10** |

Khi truyền vào biến a cho hàm incValue, thực chất trình biên dịch sẽ copy giá trị của a cho x ( x = a). Sau đó lấy giá trị đó để tính toán. Thế nên, khi chúng ta thay đổi giá trị của x, giá trị của a không thay đổi



## HÀM ĐỆ QUY

Đệ quy trong C là quá trình trong đó một phương thức gọi lại chính nó một cách liên tiếp. Một phương thức trong C gọi lại chính nó được gọi là phương thức đệ quy.

Sử dụng đệ quy giúp code chặt chẽ hơn nhưng sẽ khó để hiểu hơn.

Cú pháp:

|  |
| --- |
| < Kiểu dữ liệu trả về > < tenhamdequi > () {     // khối lệnh của hàm     tenhamdequi(); /\* gọi lại chính nó \*/  }  int main() {     tenhamdequi();  } |

Do tính chất tự gọi lại chính nó nên để tránh việc hàm đệ quy chạy không bao giờ dừng bạn luôn cần có điểm dừng (điểm dừng được hiểu đơn giản là tới 1 lúc nào đó, hàm đệ quy sẽ không gọi lại chính nó nữa).

**VD**: Cho số nguyên n được nhập từ bàn phím, bạn hãy viết hàm đệ quy trả về n! (n giai thừa).

|  |
| --- |
| #include <stdio.h>  int factorial(int n) {      if (n == 1) return 1;      return n\* factorial(n-1);  }  int main() {      int n;  printf("Nhap vao gia tri n :");      scanf("%d", &n);      printf("%d", factorial(n));      return 0;  } |

**Kết quả:**

|  |
| --- |
| **Nhap vao gia tri n :5**  **120** |

Mô tả hàm đệ quy tính 5!:

|  |
| --- |
| factorial(5)  = 5 \* factorial(4)  = 5 \* 4 \* factorial(3)  = 5 \* 4 \* 3 \* factorial(2)  = 5 \* 4 \* 3 \* 2 \* factorial(1)  = 5 \* 4 \* 3 \* 2 \* 1  = 120 |

Điểm dừng ở ví dụ trên chính là khi đầu vào của hàm factorial bằng 1 thì hàm factorial sẽ trả về 1 thay vì gọi tiếp tới chính nó.

**VD:** Cho số nguyên n được nhập từ bàn phím, bạn hãy viết hàm đệ quy trả về tổng các số từ 1 tới n.

|  |
| --- |
| #include <stdio.h>  int sum(int n) {      if (n == 1) return 1;      return n + sum(n-1);  }  int main() {      int n;      printf("Xin moi nhap gia tri n: ");      scanf("%d", &n);      printf("%d", sum(n));      return 0;  } |

**Kết quả:**

|  |
| --- |
| **Xin moi nhap gia tri n: 5**  **15** |